Design and Analysis of Algorithms
Instructor: Sharma Thankachan

Lecture 13:
Elementary Graph Algorithms III

Slides modified from Dr. Hon, with permission
About this lecture

• Depth First Search
  • Classification of Tree Edges

• Topological Sort
Classification of Tree Edges

- After a DFS process, we can classify the edges of a graph into four types:
  1. **Tree**: Edges in the DFS forest
  2. **Back**: From descendant to ancestor when explored (include self loop)
  3. **Forward**: From ancestor to descendant when explored (exclude tree edge)
  4. **Cross**: Others (no ancestor-desendant relation)
Example

Suppose the input graph is directed
Example

Suppose this is the DFS forest obtained.

Can you classify the type of each edge?
Example

Suppose the DFS forest is different now ...

Can you classify the type of each edge?
Example

Suppose the input graph is undirected
Example

Suppose this is the DFS forest obtained:

Can you classify the type of each edge?
Edges in Undirected Graph

Theorem:
After DFS of an undirected graph, every edge is either a tree edge or a back edge.

Proof: Let \((u,v)\) be an edge. Suppose \(u\) is discovered first. Then, \(v\) will become \(u\)'s descendent (white-path) so that \(f(v) < f(u)\).

- If \(u\) discovers \(v\) \(\Rightarrow\) \((u,v)\) is tree edge.
- Else, \((u,v)\) is explored after \(v\) discovered.

Then, \((u,v)\) must be explored from \(v\) because \(f(v) < f(u) \Rightarrow (u,v)\) is back edge.
Cycles in Directed Graph

Theorem: For any DFS on a directed graph \( G \), there is a back edge \( \iff G \) has a cycle

Proof:

\( \implies \) If there is a back edge \((u,v)\), it implies there is a path from \( v \) to \( u \).

Thus, this back edge completes a cycle.
Proof

(⇐) If $G$ has a cycle $C$, let

$v = \text{first vertex discovered in DFS}$

$(u,v) = v$’s preceding edge in $C$

Thus, when $v$ is discovered, all nodes in $C$ are still undiscovered (white)

$\Rightarrow v$ is ancestor of $u$ in DFS forest (why?)

$\Rightarrow (u,v)$ becomes a back edge
Topological Sort

• Directed graph can be used to indicate precedence among a set of events
• E.g., a possible precedence is dressing

- under-shorts
- pants
- belt
- shirt
- tie
- shoes
- socks
- watch
- jacket
Topological Sort

• The previous directed graph is also called a precedence graph

Question: Given a precedence graph $G$, can we order the events such that if $(u, v)$ is in $G$ (i.e. $u$ should complete before $v$) then $u$ appears before $v$ in the ordering?

We call this problem topological sorting of $G$
Topological Sort

Fact: If $G$ contains a cycle, then it is impossible to find a desired ordering
(Prove by contradiction)

• However, if $G$ is acyclic (not contains any cycle) we show that the algorithm in next slide always find one of the desired ordering
Topological Sort

Topological-Sort\((G)\) 

\{

1. Call DFS on \(G\)
2. If \(G\) contains a back edge, abort;
3. Else, output vertices in decreasing order of their finishing times;

\}

Why is the algorithm correct?
Theorem: If $G$ is acyclic, the previous algorithm produces a topological sort of $G$

Proof: Let $(u,v)$ be an edge in $G$. We shall show that $f(u) > f(v)$ so that $u$ appears before $v$ in the output ordering. Recall $G$ is acyclic, there is no back edges. There are two main cases ...
Proof

• **Case 1:** \((u,v)\) is a tree or forward edge
  
  \[ u \text{ is an ancestor of } v \]
  \[ d(u) < d(v) < f(v) < f(u) \quad \text{(why??)} \]

• **Case 2:** \((u,v)\) is a cross edge
  
  \[ d(v) < d(u) \quad \text{(otherwise, by white-path, } u \text{ must be an ancestor of } v, \text{ so that } (u,v) \text{ cannot be a cross edge)} \]
  
  \[ \text{Since } G \text{ is acyclic, } v \text{ cannot reach } u, \text{ so } d(v) < f(v) < d(u) < f(u) \quad \text{(why??)} \]

• **Both cases show** \(f(u) > f(v)\)  \(\Rightarrow\) **Done!**
Topological Sort (Example)

Discovery and Finishing Times after a possible DFS
Ordering Finishing Times (in descending order)

If we order the events from left to right, anything special about the edge directions?
Performance

• Let $G = (V,E)$ be the input directed graph.

• Running time for Topological-Sort:
  1. Perform DFS: $O(|V|+|E|)$ time
  2. Sort finishing times
     Naïve method: $O(|V| \log |V|)$ time
     Clever method: (use an extra stack $S$)
        During DFS, push a node into stack $S$ once finished $\Rightarrow$ no need to sort !!

• Total time: $O(|V|+|E|)$